View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Digital library of Brno University of Technology

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

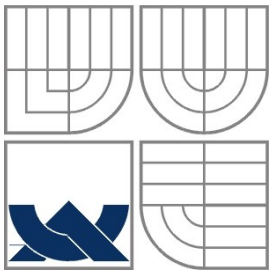
VÍCEJAZYČNÝ FOTOSERVER V RUBY ON RAILS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

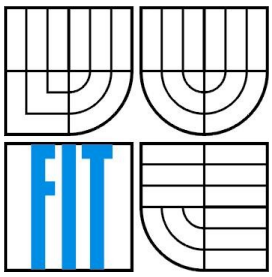
AUTOR PRÁCE
AUTHOR

LUKÁŠ KRESTA

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VÍCEJAZYČNÝ FOTOSERVER V RUBY ON RAILS

MULTILANGUAGE PHOTO SERVER ON RUBY ON RAILS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — LUKÁŠ KRESTA
AUTHOR

VEDOUCÍ PRÁCE — ING. DUŠAN VRÁŽEL
SUPERVISOR

BRNO 2008

Abstrakt

Náplní bakalářské práce bylo vytvoření vícejazyčného webového fotoalba pro více uživatelů, kde si tito můžou navzájem přidělovat práva ke svým fotoalbům. Vzhledem k použití frameworku Ruby on Rails by měl být výsledný produkt multiplatformní. Základním předpokladem pro vytvoření této práce bylo porozumění tomuto frameworku a přistoupení na konvenční programování v jazyce Ruby. Dále bylo třeba stanovit co vše bude fotoalbum umožnovat svým uživatelům a na základě tohoto navrhnout model systému, který byl následně implementován. Pro co nejsnazší přenositelnost byla aplikace implementována na databázovém systému SQLITE3, to je však možné změnit pouhým upravením konfiguračního souboru.

Klíčová slova

Ruby on Rails, FotoServer, Web, multijazyčný, Mongrel

Abstract

The goal of my Bachelor thesis work was to to create multiuser and multilanguage web photoalbum. Users can give permissions for their albums to other users. Beacause i used Ruby on Rails framework the product should be multiplatform. The basic achievement of this work was to understand this framework and use conventional programming in Ruby language. Next step was to decide what services would the photoalbum provide to users and then i designed the model which i then implemented. For the best compatibility betweet different platforms the aplication is implemented on SQLITE3 database system, this can be changed in configuration file.

Keywords

Ruby on Rails, PhotoServer, Web, multilanguage, Mongrel

Citace

Lukáš Kresta: Vícejazyčný fotoserver v Ruby on Rails. Brno, 2008, bakalářská práce, FIT VUT Brno.

Vícejazyčný fotoserver v Ruby on Rails

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Dušana Vrážela. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Lukáš Kresta
30.7.2008

Poděkování

Rád bych touto cestou poděkoval Ing. Dušanu Vráželovi za jeho odbonou pomoc při tvorbě této bakálářské práce. Poděkování patří také mému zaměstnavateli Amberg Engineering Brno a.s. za poskytnutí potřebného volného času k tvorbě tohoto projektu.

© Lukáš Kresta, 2008.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů..

Obsah

1 Úvod

Dnes, kdy již prakticky každá rodina vlastní alespoň jeden digitální fotoaparát, vzniká přirozená potřeba využít, co možná nejvíce, potenciál digitální fotografie. Za námi je rovněž období, kdy většina lidí hned po svém příjezdu z dovolené spěchala k počítači, aby co nejdříve odeslala své nejnovější snímky všem svým známým prostřednictvím hromadných emailů, nebo dokonce prostřednictvím fyzického média jako je CD a DVD. Dnes tuto potřebu sdílet naše nejkrásnější zážitky mnohem lépe naplňují služby webových fotoalb.

Těchto webových služeb k uchovávání a sdílení fotografií v digitální formě je dnes obrovské množství. Liší se ve velkém množství parametrů, především v maximálním prostoru pro jednoho uživatele, v maximální velikosti jedné fotografie, v možnosti, případně nutnosti nastavit alba jako veřejná, a tedy přístupná všem, ve spravování práv, licenčních podmínkách a dalších. Nemalý počet z těchto fotoalb nabízí rovněž online objednávku zhotovení klasické papírové fotografie z vámi již uložených předloh.

Vzhledem k této rozmanitosti stávající nabídky lze předpokládat, že vhodnou službu právě pro sebe si najde většina dnešních fotografů. Přesto dávají mnozí, především profesionálněji zaměření jedinci, přednost vlastním webovým stránkám, které tak podtrhnou svou jedinečností zároveň jedinečnost vlastníka a fotografií jím na webu zveřejněných. Jelikož patřím právě mezi ty vlastníky digitálního přístroje, kteří touží mít své dílo na vlastních webových stránkách, byl pro mě výběr tématu bakalářské práce právě z této oblasti velice lákavou záležitostí. A vzhledem k tomu, že mě v mém zaměstnání velmi zaujal framework pro tvorbu webových aplikací Ruby on Rails, požádal jsem Ing. Dušana Vrážela o vypsání tématu právě s tímto obsahem.

V úvodu druhé kapitoly se budu věnovat vývoji webové aplikace obecněji. Poté představím framework Ruby on Rails a jeho klíčové vlastnosti. Ve třetí kapitole se pak hlouběji zaměřím na prostředky pro sdílení fotografií na internetu, popíši klíčové vlastnosti takových služeb a nakonec uvedu tři významné zástupce tohoto odvětví internetových serverů. V kapitole čtvrté se zaměřím na samotnou analýzu problému vytvoření webového fotoalba, a to zcela konkrétně. Nejdříve si formálně nadefinuji požadavky na tuto aplikaci a ty následně promítnu do diagramu případů užití. Pátá kapitola se pak bude zabývat návrhem řešení aplikce s požadavky definovanými kapitolou předešlou. Hlavním produktem pak bude ER diagram návrhu databázové struktury aplikace. Popis vlastní implementace aplikace nastíní kapitola šestá. V sedmé kapitole uvedu základní charakteristiky již hotového produktu, jeho strukturou a ovládáním. Závěrečná kapitola pak shrne dosažené výsledky, přínos pro studenta a možnosti dalšího vývoje.

Autor také doufá, že bude tento text přínosný i pro čtenáře a očekává reakce na něj.

2 Tvorba webové aplikace

K tvorbě webových aplikací existuje spousta velmi odlišných prostředků a přístupů. Tím úplně nejzákladnějším je vytvoření prosté webové stránky jako statického html kódu v libovolném textovém editoru. Podobného, byť kódem poněkud „ošklivějšího“ výsledku, dosáhneme použitím nějakého speciálního nástroje na tvorbu html stránek. Mezi takové nástroje patří například Actual drawing, AceHTML Pro, Easy Web Editor a spousta dalších. Html stránky je možno v těchto aplikacích tvořit rychle a bez hlubších znalostí z této oblasti. Tento postup je pak plně dostačující například pro stránky s osobní prezentací, či pro stránky o naší kočce.

Chceme li však vytvořit plnohodnotnou webovou aplikaci, jakou je například obchod, diskuzní fórum, nebo právě webové fotoalbum, je třeba do statického html kódu doplnit některé jeho části dynamicky až za běhu aplikace. Toto se děje na straně webového serveru, kde je programová část kódu vykonána před odesláním výsledného html kódu prohlížeči. Tyto dynamicky generované části kódu bývají často odvozeny z údajů uložených v databázi. Je tedy nutné, aby daný jazyk dokázal s databázemi efektivně spolupracovat. Průkopnický jazyk, dodnes hojně používaný k dynamickému generování částí html kódu, je PHP. Tento, ač dosud nejvíce pužívaný, není ovšem zdaleka jediný. Možné je též použít jazyky jako je Perl, Java, Ruby a podobně. Záleží jen na programátorovi a webowém serveru, jaký si zvolí (většina dnešních webhostingů podporuje jen PHP).

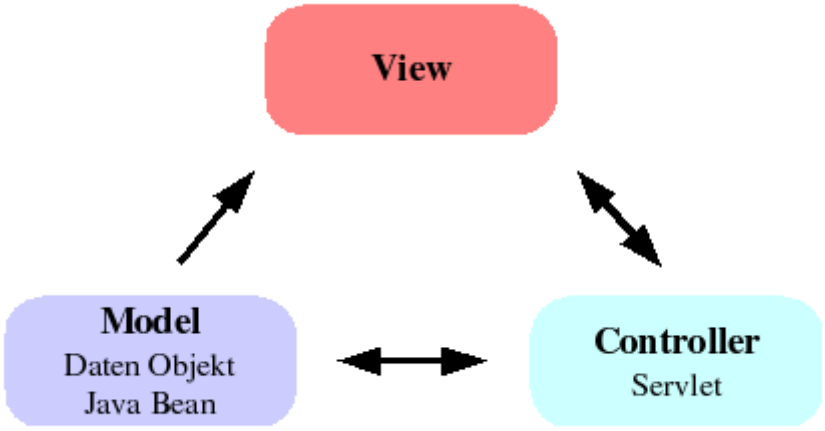
Větší webové projekty je pak vhodné implementovat pomocí vhodně zvoleného frameworku, který nám umožní efektivně oddělit jednotlivé části kódu, například pomocí některého z návrhových vzorů a zachovat tak přehlednost i pro aplikace o velkém rozsahu. Další výhodou tohoto postupu je usnadnění komunikace u týmových projektů, neboť každý programátor pracuje na vzorem oddělené části projektu.

2.1 Návrhové vzory

Návrhový vzor ve své podstatě představuje jistou formu konvence, která nám sděluje jak řešit některé problémy v různých situacích. Používá se při návrhu implementace programů. Výrazně zjednodušuje komunikaci s ostatními programátory při tvorbě týmových projektů. Objektově orientované návrhovové vzory většinou popisují vztahy mezi třídami a objekty, ale neurčují jejich implementaci. Návrhové vzory se dělí do tří kategorií: Tvořivé představují abstrakci vytváření objektů, netřeba řešit destrukci objektů; Strukturální návrhové vzory popisují složení objektů a tříd do struktur; Návrhové vzory chování definují především komunikaci mezi objekty. Tohle bych dala do odrážek navíc, když tam za těmi kategoriemi máš dvotečku, nebo z toho uděljej zvlášť věty bez středníku, ale přikláním se k odrážkám.

2.1.1 Model-view-controller

Model-view-controller (MVC, nebo též Model2) je občas prezentován jako návrhový vzor. Toto však není zcela přesné, protože se věnuje ve velké míře architektuře, a tudíž se pro něj hodí spíše označení architektonický vzor, případně agregační návrhový vzor. Hlavní myšlenkou tohoto vzoru je rozdělení aplikace na tři části, které pak jsou na sobě implementačně nezávislé. Model obsahuje reprezentace informací, s nimiž aplikace pracuje. View transformuje informace z modelů do požadované formy (většinou vizuální). Controller má za úkol reagovat na události a zpracovávat data obsažené v modelech, které pak typicky předává pohledům. Tento model je odvozen z architektury modelu-1 – ten definoval pouze dvě vrtvy, a to model a pohled spolu s řídící logikou. Jeden obecný příklad práce MVC architektury by mohl vypadat následovně: Pohled(view) přijme událost, oznámí to řadiči (controller), ten přistoupí k modelu a upraví v něm data, nakonec si pohled vyzvedne aktuální data přímo z modelu. MVC architektura se v současnosti používá především pro webové aplikace. Nejužitečnější je u větších projektů a tam, kde se často provádějí změny. Kromě Ruby on Rails využívají tuto architekturu mimo jiné například tyto webové frameworky: JavaServer Faces, CakePHP, WebObjects, ASP.NET MVC a mnoho dalších.



Obrázek 2.1. Princip architektůry MVC. Převzato z [1].

2.2 Ruby

Ruby je skriptovací, objektově orientovaný a neobyčejně přímočarý programovací jazyk. Vznikl v polovině devadesátých let minulého století na japonských ostrovech z potřeby výkonnějšího jazyka než byl v té době Perl, avšak při zachování objektové orientace. Ta je v případě Ruby prakticky

dokonalá, poněvadž v tomto jazyku je objektem úplně vše. Dále je zde kladen velký důraz na možnost psaní kódu co nejčitelněji. To v sobě mimo jiné zahrnuje možnost konstrukce kódu, který co nejvíce připomíná lidský způsob uvažování. Velkou výhodou je přenositelnost výsledného kódu, a tím pádem nezávislost na platformě. Zvláště při seznamování se syntaxí spousta lidí ocení možnost spuštění kódu v interaktivním režimu prostřednictvím programu IRB, který příkazy interpretuje ihned po jejich odřádkování. Snad jedinou nevýhodou Ruby je jeho rychlost, která je vzhledem k překladu až za běhu programu o dost nižší než je tomu u kompilovaných jazyků jakými jsou například C a Java.

Nyní sílu a klíčové vlastnosti Ruby demonstruji na pár příkladech a v některých srovnám výslednou konstrukci se stejným kódem v jiném jazyku. V Ruby není například žádný problém vrátit z funkce nebo výrazu dvě proměnné při jednom volání. Kód takového výrazu může pak vypadat například takto: `a, b = b, a % b`

Tímto bude proměnné `a` přiřazena hodnota proměnné `b` a proměnné `b` zbytek hodnoty v proměnné po celočíselném dělení hodnotou `b`. Tato funkcionalita je velmi užitečná především pro vracení více poměnných z jedné funkce, kde tak programátorovi odpadá nutnost volat funkci vícekrát, předávání výsledku ukazatelem, či často ne zcela bezpečné používání globálních proměnných.

Další velice zajímavou inovací jazyka je možnost napsat podmínku až za příkaz, či blok příkazů, který je podmíněn. Při vhodném užívání této možnosti může dojít k značnému zvýšení přehlednosti výsledného kódu. Příkladem budiž následující konstrukce: `puts 'Vyhrál jsem' if tip == los`.

V tomto případě se tiskne řetezec 'Vyhrál jsem' jen za předpokladu, že tip je shodný s losem. Proměnné *tip* a *los* uvažujeme v tomto případě jako celočíselné proměnné. Naproti tomu využitím unless dosáhneme vykonání podmíněného kódu jen v případě, když podmínka není splněna. Ukáži to na obdobném příkladu: `puts 'Prohrál jsem' unless tip == los` Tento kód je pak defakto identický k následujícímu: `puts 'Prohrál jsem' if tip <> los`, tedy řetězec 'Prohrál jsem' je tisknut v případě, kdy se tipované a losované čísla neshodují.

Vzhledem k plné objektovosti je možné pracovat jako s objekty i s čísly, řetezci a podobně. To přináší řadu výhod, které demonstruji na následujících příkladech: -123.abs vrátí absolutní hodnotu objektu s hodnotou -123 a tedy kladné číslo 123. Nebo můžeme vytvořit pole retpol = ['xxx', 'yyy'] a poté zavolat metodu last třeba takto: puts pole.last Tato metoda nad námi vytvořeným objektem retpol vrátí hodnotu svého posledního prvku funci puts, a ta tento výsledek vytiskne. Bude se tedy tisknout řetězec 'yyy'.

Podle mého názoru je jednou z nejužitečnějších inovací jazyka Ruby metoda times, která se volá nad celočíselnými objekty. Této metodě je jako parametr předán blok kódu, který má být vykonán, právě tolikrát, jaká je právě hodnota tohoto objektu. V některých jazycích ukáži

implementaci části kódu, která výhodu a čitelnost konstrukce s využitím funkce times objasní zřetelněji. Úkolem je vytisknout prvních deset sudých čísel od nuly

```
Python: for i in range(10):
            print i
 Ruby: 10.times |i|
            print i
```

Jsem přesvědčen, že zápis „`10.times`“ je daleko čitelnější než různé konstrukce pomocí cyklu for, či jiných.

Dnes je Ruby v České republice používáno především jako součást frameworku Rails. Myslím si ale, že je škoda, že se v jiných oblastech zatím moc neprosadilo, protože se dle mého názoru jedná o nejpřímočařejší a nejčitelnější programovací jazyk poslední doby.

2.3 Ruby on Rails

Je několik užitečných knihoven shrnutých do rozsáhlejšího frameworku sloužícího pro rychlé, bezpečné a přehledné vytváření plnohodnotných webových aplikací. Ten je pak celý postaven na přímočarém a moderním plně objektovém skriptovacím jazyku Ruby, což znamená, že v něm nejen programátor píše veškerý svůj kód k interpretaci, ale jsou v něm implementovány i veškeré knihovny tohoto frameworku. Jedinou nevýhodou tohoto je pak zvýšený požadavek na výpočetní výkon serveru, což v dnešní době nepředstavuje pro většinu projektů podstatnější problém, takže popularita Ruby on Rails (dále jen RoR) neustále roste.

Pravdou je, že použití frameworku RoR zvýší produktivitu práce programátorů až několikanásobně. RoR totiž přebírá mnohé klasické problémy vývoje webových aplikací pod svá křídla a programátor se tak může plně soustředit na implementaci vlastní užitečné funkcionality webu. Mezi základní úkoly převzaté Railsy za své patří komunikace s databází. Tu plně převezme datová struktura v modelu, kterou zajistí modul ActiveRecord. Tím nám poskytne plně objektovou abstrakci databáze a mnohem jednodušší přístup k ní. Vše, co je pro to nutné udělat, je jednoduše nakonfigurovat soubor database.yml v struktuře Rails projektu. Ale dále se k vytvoření a konfiguraci projektu vrátím až v kapitole popisující vlastní implementaci mé bakalářské práce.

RoR jsou postaveny na softwarové architektuře Model-View-Controller. Tato rozděluje vyvíjenou aplikaci na tři části. Model představuje abstrakci datových struktur, které v sobě uchovávají informace, se kterými aplikace pracuje. V případě RoR model přebírá ještě i funkci objektové reprezentace databáze a přes něj lze s databází velmi jednoduše a efektivně komunikovat. Controller obsahuje interpretovatelný Ruby kód, kterým reaguje na události z objektů typu View. Pracuje s datovými abstrakcemi obsažených v modelech a poskytuje výstup zpět do View. View představují formu prezentace výstupních dat uživateli. V případě RoR se tedy jedná o šablony html

stránek,
v nichž jsou vloženy informace o objektech obsažených v modelech a výstupy z Controlleru.

Dále je zde třeba zmínit, že RoR preferují konvence před syntaxí. To prakticky znamená, že je v nich třeba dodržovat určitá pravidla zejména pro názvosloví většiny součástí projektu. Jedním z nejtypičtějších příkladů je název tabulky v databázi. Její název by podle konvence měl být v množném čísle podle anglického vzoru. Ne že by nebylo možné použít český název, ale rozhodně by to neměla být kupříkladu tabulka *POLOŽKY*! Místo toho ji pojmenujeme *POLOZKAS*, přistupovat k ní pak budeme přes model, který ponese její název v jednotném čísle, první písmeno velké.

2.3.1 Práce s ActiveRecord

ActiveRecord nám poskytuje abstrakci k databázovému systému. K datům pak můžeme přistupovat skrz něj zcela objektově. To si ukážeme na následujících příkladech.

```
@albums = Album.find(:all, :conditions => [ "user_id = ?",
session[:user].id])
```

najde všechny položky z tabulky albums které splňují podmínku a uloží je do hashe `@albums`.

```
@photo = Photo.find(params[:id])
```

Do proměnné `@photo` řádek tabulky photos s id obsaženém v `params[:id]`

3 FOTOSERVERY

Webových služeb k uchovávání a sdílení fotografií v digitální formě je dnes obrovské množství. Kromě vzhledu uživatelského prostředí se od sebe liší především maximálním prostorem pro jednoho uživatele, maximální velikostí jedné fotografie, možností, případně nutností nastavit alba jako veřejná a tedy přístupná všem, spravováním práv, licenčními podmínkami, způsobem uploadu fotografií na server, možností přímé editace přes webové rozhraní. Nemalý počet z těchto fotoalb nabízí rovněž online objednávku zhotovení klasické papírové fotografie z vámi již uložených předloh.

Na následujících řádcích bude následovat popis některých z těchto služeb. Budu se snažit vystihnout jejich klíčové, kladné a záporné vlastnosti, a taky pro jakou skupinu uživatelů se která služba hodí. Z předchozího odstavce plyne, že vlastností takového webového fotoserveru je příliš na to, abych zde mohl hodnotit všechny. Proto budu vybírat jen ty, které jsou z mého pohledu klíčové.

3.1 RAJCE.NET

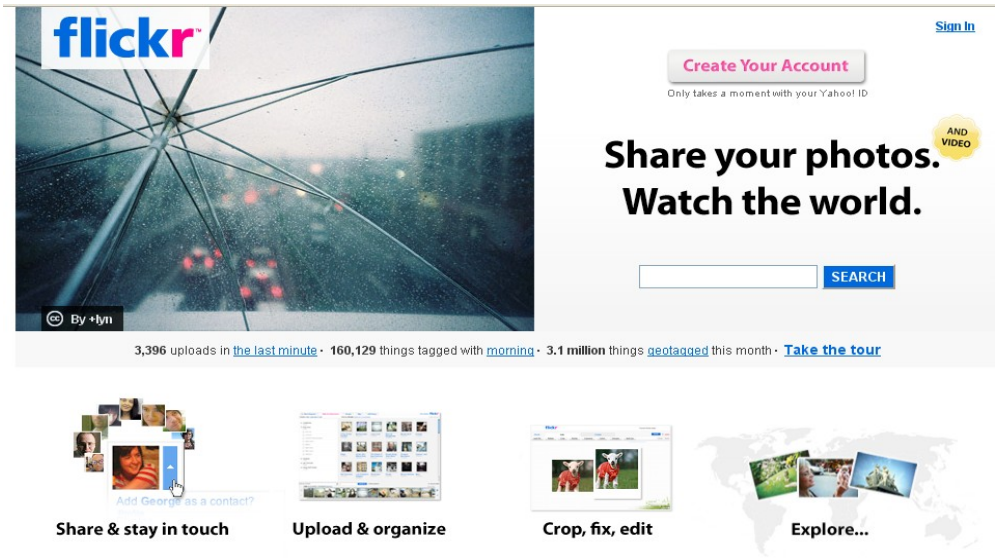
Obrázek 3.1. Úvodní stránka rajce.net. Převzato z [2]

Jedná se o v českém prostředí jednu z nejznámějších a nejpoužívanějších služeb. V současné době je v rodině služeb serveru iDnes.cz. Server sám sebe hrdě označuje jako největší fotoalbum na českém internetu. Své zákazníky láká především kvalitním uživatelským rozhraním v češtině, jednoduchou webovou adresou na prohlížení vlastních fotografií ve tvaru přezdívka.rajce.net, vcelku jednoduché nahrávání fotografií na server pomocí kvalitního softwaru, možností uložení neomezeného počtu fotografií, a to zdarma.

Mezi největší nevýhody tohoto serveru patří bezesporu naprostá absence možností přidělování práv k jenotlivým fotoalbům. To je zde alespoň částečně kompenzováno jednak zneviditelněním (k albumu se pak dostanou jen lidé znající jeho přesný název), a jednak možností zamknutí alba heslem – ovšem představa, že budu po každém uploadu posílat emailem zvoleným přátelům stále nová a nová hesla, moc lákavá není. Další, ne příliš pozitivní vlastností je snížení rozlišení (a tedy i kvality) snímku hned při jeho uploadu na server. Možnost přidat alespoň základní komentář k fotografii nebo alespoň albu by také některé uživatele jistě potěšila.

Jak tedy plyne z předchozího, tato služba je velmi jednoduchá k používání a hodí se například ke sdílení fotografií z podnikového večírku, kde budou všichni kolegové nadšeni v jak krásné náladě jste je to na svůj mobilní CCD snímač zrovna zachytili. Naopak je tato služba naprosto nevhodná jako jeden ze způsobů zálohy svých nejkrásnějších kousků, či k udržování určité komunity lidí (přátel, rodina) a jejich vzájemnému sdílení vybraných alb.

3.2 FLICKR.COM



Obrázek 3.2. Úvodní stránka flickr.com . Převzato z [3]

Nyní se pro změnu zaměřím na službu populární především ve světě. Jedná se o webový fotoserver patřící významnému hráči na poli světového internetu – společnosti YAHOO. V tom je první výhoda této služby, avšak jen pro některé. Jste-li totiž uživateli některé z jiných služeb této společnosti, nemusíte se již registrovat, použijete přihlašovací údaje, které už máte přiděleny právě k oné jiné službě. To ovšem neznamená, že podstoupit registraci nového uživatele je něco složitého. Naopak patří mezi ty kratší a s jejím zvládnutím by neměl mít nikdo žádné větší potíže. Ihned po registraci je možné se přihlásit a začít s uploadem fotografií na server. To je možné přes webové rozhraní, nebo pomocí jednoduchého a příjemného prográmku FlickrUploader. A co hodnotím jako velký klad - tento program je k dispozici pro platformy Windows, Linux, i MAC OS X. V momentě, kdy fotografii nahráváte na server si sami určíte, zda je veřejná, či privátní. V případě privátní pak ještě můžte určit které skupiny (rodina, přátelé) uvidí fotografie spolu s vámi.

Služba se dělí na dvě varianty - placenou a zdarma. U verze bez poplatku jste omezení limitováni maximálním uploadem 100MB nových fotografií za měsíc. Za roční poplatek 24.95 USD je prostor pro vaše fotografie neomezen. Další výhodou placené verze je možnost nahrání fotografie jakékoliv velikosti. Vaši přátelé pak uvidí skvosty poslední dovolené v celé své parádě. V případě verze bez poplatku je tato velikost automaticky omezena na 1024x768 pixelů.

V současnosti umí Flickr osm jazyků, češtinu však mezi nimi nenajdete. To je však spolu s poněkud méně přehledným uživatelským prostředím jediná drobná vada na kráse této služby. Pro uživatele, kteří mají základy angličtiny, fotí víc než pouze mobilem a jsou ochotni zaplatit 25 USD ročně, je toto fotoalbum velmi výhodné. Pro ostatní – skromnější uživatele, je také nezpoplatněná verze dostatečně kvalitním řešením.

3.3 IMAGESHACK.US

Obrázek 3.3. Úvodní stránka imageShack.us . Převzato z [4]

Tato služba se ani fotoalbem nazvat nedá, přesto ji zde zmiňuju, protože se může v některých případech jednat o pro mnohé ideální řešení. Služba totiž umožňuje bez jakékoliv registrace nahrát fotografii až do velikosti 1.53MB. K té se ihned po uploadu zobrazí hypertextový odkaz který lze poslat přátelům a podobně. Jedná se tedy o vynikající možnost jednorázově sdílet maximálně pár jednotek fotografií. Nevýhodou je omezená velikost fotografie a ne příliš pěkná výsledná adresa fotografie. První nevýhoda je částečně vykompenzován možností automatické úpravy velikosti tak, aby se do limitu vešla. Naopak pozitivní stránkou této služby je to, že obrázky nejsou to jediné co lze jejím prostřednictvím sdílet.

4 Analýza

Před návrhem vlastního řešení webového fotoalba si musíme určit, jakou funkcionalitu od výsledného produktu očekáváme. Z analýzy této požadované funkcionality pak sestavíme model případů užití, z nějž budeme vycházet při tvorbě vlastního návrhu řešení. Analýza vychází jak ze samotného zadání projektu, tak z požadavků potencionálních uživatelů a průzkumů chování ostatních, podobných produktů na dnešním internetu. Tato kapitola vychází především z [5]

4.1 Funkční požadavky

Základní vlastnosti implementovaného projektu jsou dány již zadáním bakalářské práce. Album by mělo mít privátní charakter – pro jeho užívání je tedy třeba se nejdříve zaregistrovat a poté přihlásit.

Přihlášený uživatel pak musí mít možnost vytvářet svá fotoaba i s vlastním komentářem a plnit je fotografiemi. Ty si pak smí dle libosti prohlížet, editovat názvy, či je z fotoalba odstranit. K fotografiím dále smí přidávat komentáře, editovat je a mazat, komentáře ostatních uživatel může u svých fotografií pouze odstranit. Ke svým vlastním albům smí přidělovat práva ostatním uživatelům, či je činit veřejnými, tedy k prohlížení dostupné všem.

Dále si může prohlížet alba, k nimž mu přidělili práva jeho přátelé. Podle stupně oprávnění pak může i přidávat komentáře k fotografiím, stahovat si je v plném rozlišení, přidávat nové fotografie, či naopak fotografie mazat. V případě práva komentovat může své komentáře samozřejmě i mazat, či editovat. Přístup má i ke všem albům veřejným, ty si však (pokud nemá zároveň přidělená práva od vlastníka explicitně) může jen prohlížet.

Uživatel má také možnost upravit si v nastavení jazyk, ve kterém bude s uživatelským rozhraním komunikovat a vizáž pomocí předdefinovaných skinů. Množství podporovaných jazyků i vzhledů by se mělo v budoucnu rozrůstat. Při implementaci je tedy třeba postupovat tak, aby se nová data do slovníku dala přidávat co nejefektivněji a nový skin třeba pouhým dodáním souboru se stylem a následnou úpravou jedné tabulky.

4.2 Nefunkční požadavky

Nefunční požadavky nebyly v zadání nikterak definovány, přesto jsou některé z nich velmi jasné již samotným zaměřením aplikace. Pokusím se ty nejdůležitější shrnout v následujícím seznamu.

- Uživatelské rozhraní musí být jednoduché na ovládání a plně intuitivní.
- Jednotlivé části kódu by měly být co nejuniverzálnější, aby se daly využít i u jiných projektů.
- Kód by měl být čitelný a lehce upravitelný jiným programátorem.
- Aplikace musí být řádně otestována a před nasazením do ostrého provozu by měly být odstraněny veškeré závažné chyby.
- Při výkonech současných webových serverů by na jejich straně problém s výkonem nastat neměl, přesto by měl být při implementaci kladen důraz na efektivnost výsledného kódu.

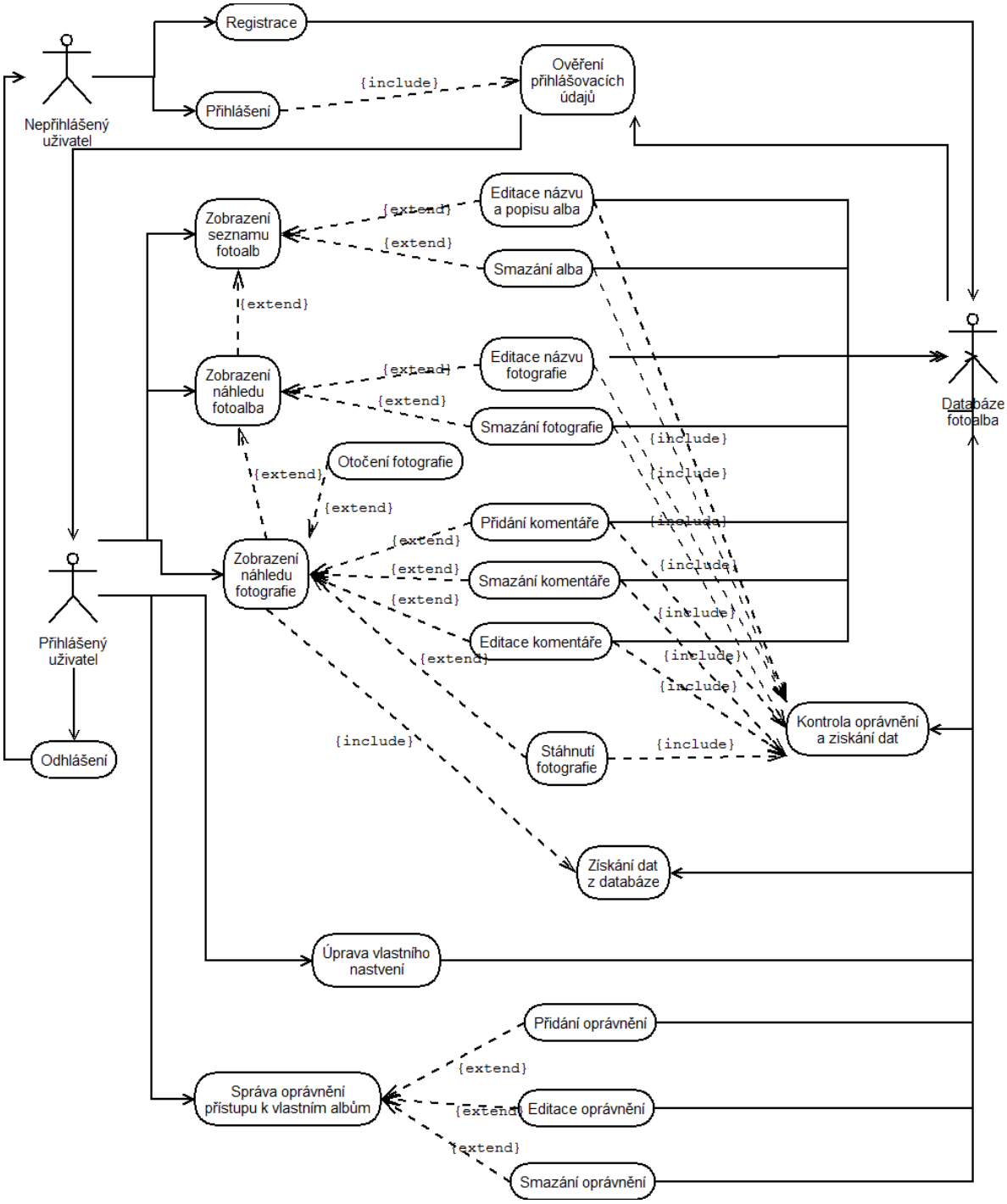
- Obrázky musí být před odesláním klientovi transformovány na zobrazovanou velikost, aby se zbytečně nezatěžovala linka.

- Měly by být uplatněny co nejefektivnější programátorské postupy. Ušetřené hodiny na implementaci je vhodné investovat do testování výsledného produktu.

- Systém musí být i po ukončení vlastní implementace spravován a pravidelně aktualizován s ohledem na požadavky uživatelů.

- Velmi vhodné by bylo zachovat plnou přenositelnost serveru na jiné operační systémy. Jedná se především o používání vhodných multiplatformních knihoven v naší aplikaci. Ruby on Rails plně přenositelné jsou.

4.3 Graf případů užití



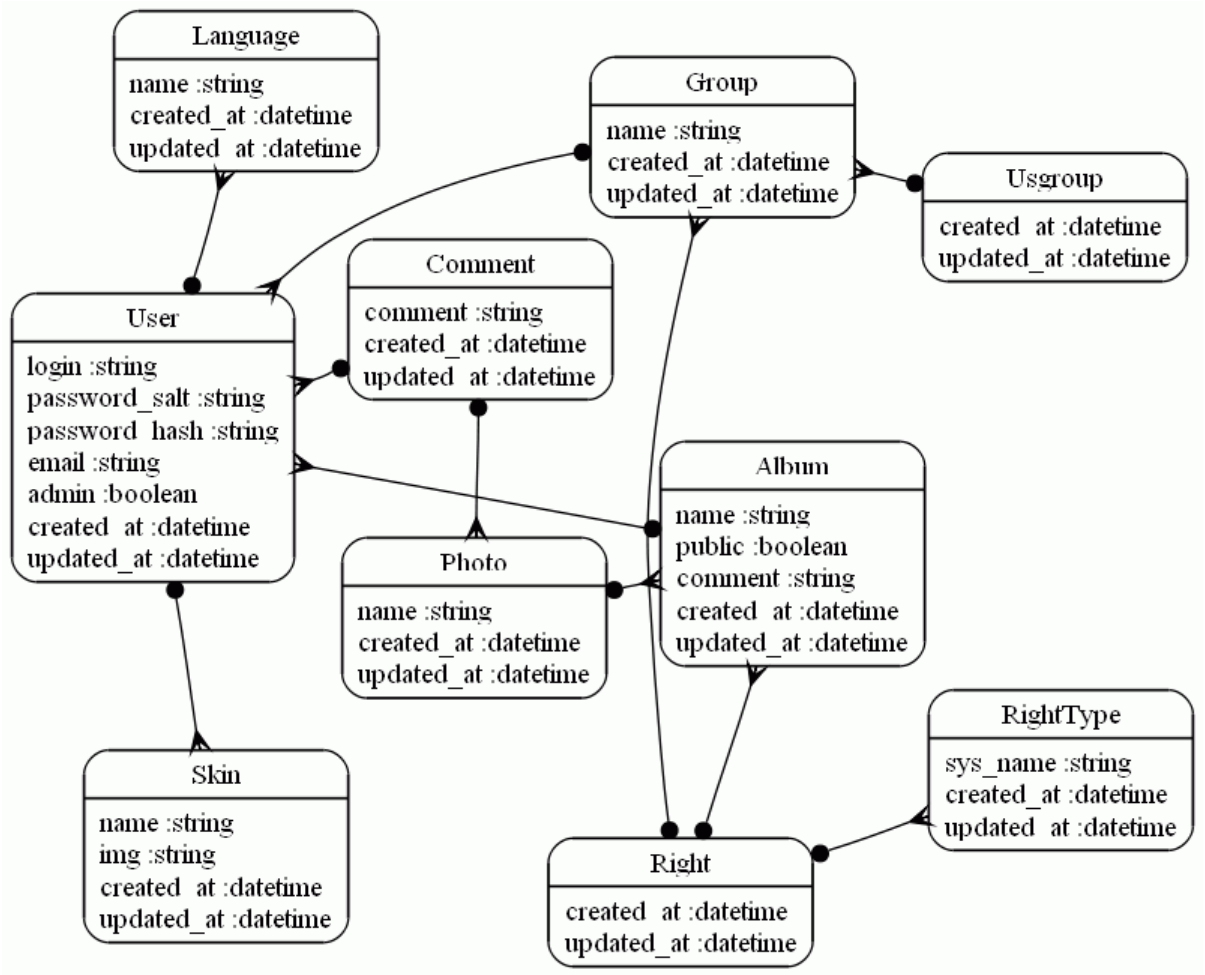
Obrázek 4.1. Graf případů užití

5 Návrh

Tato část vychází z velké části z kapitoly předešlé, kde jsou definovány hlavní požadavky na výsledný produkt. Hlavním úkolem zde je navrhnout databázový systém, který by umožňoval efektivně spravovat všechny důležité informace o uživatelých, jejich fotoalbech, fotografiích, právech

a komentářích. Výsledkem je databáze s deseti tabulkami, reprezentovaná následujícím ER diagramem.

5.1 ER diagram



Obrázek 5.1. ER diagram aplikace

Diagram je generován z modelů, tudíž jsou zde tabuky pojmenovány jednotným číslem. Tabulka *users* obsahuje informace o registrovaných uživatelých – především jejich přihlašovací údaje. *Albums* obsahuje údaje o albumech všech uživatelů a je provázaná s tabulkou *photos*, která shraňuje informace o *fotografiích*. V *comments* jsou uloženy komentáře uživatelů k fotografiím. Do tabulky *rights* se ukládají práva skupin uživatelů. Tyto skupiny se pak nacházejí v tabulce *groups* a aktéři jsou do nich přiřazeni skrz tabulku *usgroups*. Tabuka *right_types* obsahuhuje výčet typů práv, které mohou vlastníci na své alba poskytovat. V tabulce *languages* jsou pak názvy všech podporovaných jazyků k překladu stránek. Vzhled aplikace uživatel vybírá z nabídky v tabulce *skins*.

6 Popis implementace FotoAlba

Implementace webového FotoAlba je rozdělena do několika na sobě přímo nezávislých celků. Jeden se stará a práci s uživately, jiný například o fotoalba. Každý z těchto celků pracuje nad architekturou MVC, a je tedy přehledně rozdělen do tří implementačně na sobě nezávislých částí. Z tohoto důvodu je celá aplikace přehlednější a změny v ní jednodušší, než je tomu například u PHP. Při vývoji bylo třeba respektovt konvence RoR a občas přistoupit na jeho v jistém smyslu ne zcela programátorské postupy. V momentě, kdy se však s Railsy lépe seznámíte, nebudete se již zřejmě chtít, stejně jako mnozí mí kolegové, ke „klasickému“ programovní již nikdy vrátit. Jako poslední bych v úvodu této kapitoly zmínil, že jsem při vlastním programování z velké míry vycházel z [6].

6.1 Instalace frameworku RoR

Nejdříve je třeba stáhnout instalátor Ruby. V mém případě se jednalo o doporučenou verzi Ruby 1.8.6 určenou pro operační systém windows. Hned na to je třeba nainstalovat k ruby balíček RubyGems. Ten je v době psaní tohoto textu k dispozici ve verzi 1.2.0. Poté již můžeme jednoduše online instalovat další rozšíření pomocí gem balíčků. Začneme tedy tím nejdůležitějším – spustíme konzolu příkazové řadky a příkazem

```
gem install rails --include-dependencies
```

spustíme instalaci nejnovější verze RoR. Celá instalace by měla proběhnout během pár minut a zcela automaticky, maximálně budete občas dotázáni na potvrzení instalace některého z balíčků kvůli jeho závislosti k Railsům. Po úspěšném nainstalování rámce Rails na jazyk Ruby můžeme směle začít s vlastní implementací své webových aplikací.

6.2 Instalace a nastavení vývojového prostředí

Během vlastní implementace projektu jsem vyzkoušel více vývojových prostředí. Začal jsem u programu Eclipse, který používáme při vývoji RoR aplikace i v zaměstnání. Během vývoje mi bylo ale ukázáno prostředí NetBeans pro RoR. Ze zvědavosti jsem jej tedy nainstaloval na svůj osobní notebook a pokračoval v práci. Uživatelské prostředí NetBeans mi subjektivně přišlo mnohem přívětivější než prostředí Eclipse. Ani u tohoto prostředí jsem však příliš dlouho nezůstal a v touze poznat ještě něco nového jsem se odhodlal nainstalovat ještě produkt Aptana Studio. V tomto prostředí, které mi přišlo Eclipsu velmi podobné, jsem nakonec celý svůj projekt dokončil, a proto popíšu práve jeho isnstalaci a nastavení pro RoR.

Ze stránek www.aptana.com jsem si stáhnul nejnovější verzi produktu Aptana Studio a nainstaloval ji. Modul RadRails by již měla obsahovat, takže jsem ji spustil, vybereme výchozí

složku, ve které budeme produkty vyvíjet a přepneme do módu, který umožňuje pracovat s RoR projekty. To uděláme kliknutím na tlačítko Open perspektive a vybráním RadRails perspektive.

Nyní je čas prostředí Aptana Studio nastavit pro práci s RoR na našem počítači. Přejdeme tedy v menu do záložky Window a tam zvolíme Preferences, tedy nastavení aplikace. V dialogovém okně přejdeme do záložky Rails, Configuration a vyplníme zde cestu k binárnímu souboru rails a mongrel_rails. Potvrdíme a následně přejdeme do záložky ruby, kde v složce rake doplníme adresářovou cestu k tomuto binárnímu souboru. Opět potvrdíme a dialog Preferences můžeme zavřít. Tímto je příprava aplikace k vývoji RoR aplikcí dokončena. Další nastavení již záleží ne chuti každého programátora.

6.3 Vytvoření základu aplikace

Nejdříve ze všeho je dobré provést nastavení vlastního databáazového systému, který ve spojení s naším produktem budeme využívat. Ve druhé verzy RoR je primárně nastavena vlastní databáze na SQLite3. To je defakto souborová databáze, což s sebou přináší obrovskou výhodu v možnostech přenesení na jiný počítač či server. Databáze totiž, stejně tak jako její ovladač, může být přímo součásti adresářové struktury projektu. Pokud tedy na jiný systém zkopírujeme náš projekt, zkopírujeme s ním i databázi. Neshledal jsem tedy žádný důvod hledat jako databázový server jinou volbu.

Přesto není žádný problém přejít na jakýkoliv jiný běžný databázový sever. Stačí jen doinstalovat ovladač (například příkazem gem install mysql) a změnit nastavení v konfiguračním souboru database.yml. Tímto jsme s konfigurací databáze hotoví, o ostatní se postarjí Railsy samy. Následuje příklad konfigurace databáze v souboru database.yml, z něj jasně vyplývá, že není problém použít dva druhy databáze pro různé módy běhu serveru:

```
development:
  adapter: sqlite3
  database: db/development.sqlite3
  timeout: 5000

# Warning: The database defined as 'test' will be erased and
# re-generated from your development database when you run 'rake'.
# Do not set this db to the same as development or production.
test:
  adapter: mysql
  database: testova
  timeout: 5000
username: radimek
password: otik43
```

```
production:
  adapter: sqlite3
  database: db/production.sqlite3
  timeout: 5000
```

Jsme připravení začít se skutečným vývojem aplikace. Začneme vytvořením základní kostry aplikace. Ta není zrovna jednoduchá, protože oobsahuje nemalé množstvi konfiguračních souborů, konvencí stanovenou strukturu složek a spoustu dalších souborů o většině z nichž nemusíme při vývoji projektu prakticky nic vědět. Kostru vytvoříme buďto z příkazové řádky zavoláním rails, kterému jako parametr předáme název aplikace. V našem případě by se tedy zřejmě jednalo o příkaz „rails PhotoAlbum“. Ale když již máme otevřené prostředí Aptana Studio, klikneme na File, New a zde vybereme Rails projekt. Vyplníme jméno, zvolíme typ databáze, který bude naše aplikace používat a server, na kterém poběží. Potvrdíme a Railsy nám vygenerují strukturu nového projektu. V záložce Servers můžeme ihned spustit a otestovat námi vytvořený webový server – osobně preferuji mongrel před webrickem, a to čistě ze subjektivních důvodů.

6.4 Vytvoření lešení aplikace

Nyní, když máme vytvořenu základní strukturu projektu, můžeme začít tvořit triviální programové jednotky, které nám rozdělí výslednou aplikaci na více implementačne nezávislých programových jednotek. Toho nejjednodušeji dosáhneme použitím generátoru *scaffold*, který je přímo součástí RoR. Přejdeme tedy ve spodní liště do záložky *Generators* a z nabídky vybereme *scaffold*. Nyní budeme postupovat podle návrhu a ER diagramu. Jako první vytvoříme „lešení“ pro fotoalba. Do řádku parametrů tedy zapíšeme název modelu v jednotném čísle a jména sloupců v tabulce, které budeme chtít vytvořit spolu s jejich datovým typem. Jména jsou oddělena mezerou a za nimi následuje datový typ oddělený dvojtečkou. V našem případě by parametry byly náledující:

```
album name:string user_id:integer public:bolean comment:string
```

Sloupec ID se vytváří automaticky. Po potvrzení nám scaffold pro album vytvoří c*ontroller, helper, model, view a migraci*. Podobně vytvoříme „lešení“ i pro tabulky *photos, comments, groups a rights*.

6.5 Vytvoření databáze

Zde vytoříme migrace i pro ostatní tabulky, které budeme v naší aplikaci potřebovat. Konkrétně tedy pro tabuky *users, right_types, languages a skins*. Přejdeme zpět do záložky generators, vybereme migration a jako parametr doplníme jméno tabulky – tentokrát již v množném čísle. Ve struktuře projektu vyhledáme generátorem nově vytvořené migrace (nacházejí se ve složce db a podsložce migrate) a upravíme je tak, aby odpovídaly návrhu naší databáze. Například soubor *005_create_users.rb* by mohl vypadat následovně:

```
class CreateUsers < ActiveRecord::Migration
  def self.up
    create_table :users do |t|
      t.string :login
      t.string :password_salt
      t.string :password_hash
      t.string :email
      t.integer :skin_id
      t.integer :language_id
      t.boolean :admin
      t.timestamps
    end
  end

  def self.down
    drop_table :users
  end
end
```

Máme-li všechny migrace přípraveny, můžeme přikročit k samotnému vytvoření databáze. Přejdeme tedy do záložky *Rake Tasks*, z nabídky vybereme *db:create:all* a potvrzením vytvoříme databázi pro každý mód běhu webového server. Databáze jsou sice vytvořeny, ale zcela prázdné. Proto spustíme ještě jeden *Rake Task* a to *db:migrate*. Ten zajistí vytvoření databázové struktury podle modelu, který jsme definovali v migračních souborech. Za parametr r*ails_env*= přidáme název módu databáze, kterou chceme migrovat. Nejčasteji tedy v základu vytvořené módy *test, production*, či *development*, který je zároveň defaultním. Po úspěšné migraci je databáze připravena a můžeme se zaměřit na tvorbu samotného kódu.

6.6 Nastavení závislostí v modelech

V jednotlivých modelech můžeme nyní nastavit závislosti k udržení referenční integrity databáze. To učiníme pomocí klíčových slov has_many a belongs_to. Asi už není třeba zdůrazňovat, že za prvním následuje název v množném čísle, kdežto za druhým v jednotném. K určení každé závislosti je možné ještě parametrům přidat určitá pravidla. V mém případě se jedná zejména o `:dependent => :destroy`, které zaručí zničení všech závislých odkazů referované tabulky při smazání záznamu. Například model photo.rb bude poté vypadat následovně:

```
class Photo < ActiveRecord::Base
  belongs_to :album
  has_many :comments, :dependent => :destroy
end
```

6.7 Přihlašovací systém

Přihlášení do fotoalba patří k naprosto nejzákladnější funkcionalitě. Proto jej zpracujeme jako první. K RoR existuje na internetu velké množství generátorů, které vytvoří přihlašovací systém za programátora. Mezi nejznámější patří login generator, který můžeme nainstalovat online pomocí příkazu `gem install login_generator`. Já se však rozhodl, že si přihlášení naimplementuju ručně postupem popsaným na stránkach [7]. Minimálně tím získám o něco větší povědomí, než jsem dosud měl.

Nejdříve si vytvořím *Controller Login*. Poté připravím ve *Views* formuláře přihlašovacího okna a formulář pro registraci nového uživatele. Do modelu *user.rb* pak doplním funkci *password*, která vygeneruje náhodný textový řetězec jako položku *password_salt*, a pomocí tohoto řetězce pak zákóduje heslo pomocí šifry MD5 do pole *password_hash*. Této funkce využijeme při registraci nového uživatele, heslo tak bude lépe chráněné.

```
def password=(pw)
    salt = [Array.new(6){rand(256).chr}.join].pack("m").chomp
      #vytvoříme náhodný řetězec
    self.password_salt, self.password_hash =
      salt, Digest::MD5.hexdigest(pw + salt)
      #a tím zakódujeme heslo do databáze
  end
```

Další funkci, kterou budeme potřebovat při přihlášení uživatele, je *password_is*. Ta nám ověří správnost použitého heslo. Do modelu přidáme ještě validaci unikátnosti přihlašovacího jména a potvrzení hesla pro registraci nového uživatele. Funkce z modelu pak použijeme v *login controlleru*, který navážeme na příslušné formuláře. Při úspěšném přihlášení si pak do session uložíme informace o uživateli, včetně jeho preferovaném jazyku a grafickém prostředí.

```
def password_is?(pw)
    Digest::MD5.hexdigest(pw + password_salt) == password_hash
      #Funkce zakóduje příchozí heslo stejným řetězcem jako v případě
      #uložení, výsledek se tedy musí rovnat uloženému zašifrovanému heslu
  end
```

6.8 Překlad jazyka stránek

Tou zcela nejprimitivnější metodou, jak učinit své stránky multijazyčné, je bezesporu zrcadlení, neboli plnohodnotná kopie stránek, pouze každá s textem v příslušném překladu. Server pak vybere na základě požadavku klienta, kterou stránku v jaké jazykové mutaci odešle. Tento způsob překladu se zcela jistě nehodí pro větší projekty. Složitost administrace textů by byla velice obtížná.

Mnohem elegantnější způsob je přiřazení všech na webové stránce zobrazovaných textů speciálně vytvořené metodě, která na jejich místo vrátí textový řetězec v jazyce daném nějakou globální proměnnou. Tento způsob jsem zcela logicky zvolil. Po krátkém pátrání na internetu jsem jako nejvhodnější prostředek pro překlad mých stránek zvolil plugin gettext ve verzi 1.8.0. Při jeho instalaci jsem se řídil instrukcemi v [8]. Všechny řetězce určené k překladu přiřadíme metodě, jejíž název je jediný znak, a to podtržítko. Z řetězce 'hello' tak uděláme _('hello') a podobně.

Dále bylo třeba připravit si dva Rake tasky. Prvním byl updatepo, který po svém spuštění vyhledá v projektu řetězce k překladu a naplní jimi slovníky. Ty jsou v .po formátu a k jejich pohodlnému překladu se dá jednoduše použít program poEdit, dostupný z [9]. Poté co příslušné slovníky přeložíme, či upravíme, spustíme druhý task – makemo. Ten zajistí vytvoření .mo souborů, které obsahují databázi textových řetězců s překlady.

Poté již jen jednoduše přepínáme požadovaný jazyk prostřednictvím makra *set_locale*, nebo přímo nastavením jazyka v Cookies.

6.9 Změna vzhledu uživatelského prostředí

Zde lze podobně jako v předchozím případě udělat více webových stránek stejného obsahu, avšak rozdílné vizáže. Toto řešení mi však připadne velmi neelegantní. Druhou možností je generovat části kódu reprezentující vzhledové vlastnosti webové stránky dynamicky podle zvoleného vzhledu uživatele. Zvolil jsem tedy druhou variantu - konkrétně dynamické vložení .css souboru s předdefinovanými styly pro skin. Ten si uživatel zvolí již během své registrace, kdy všechny dostupné skiny jsou pod svým jménem přístupné z tabulky skins. Tento skin si pak ale může kdykoliv změnit ve svém nastavení.

6.10 Zobrazení fotografie

Vzhledem k možnosti náhrávat a archivovat na serveru fotografie neomezené velikosti je nutné je před odesláním webovému prohlížeči k náhledu ve webové stránce ořezat na zobrazovanou velikost. K tomuto je třeba pomocí nějakého rozhraní použít externí nástroj. Já se rozhodl pro program

ImageMagick, který znám ze zaměstnání. Vím o něm, že má velmi rozsáhlé možnosti co se týče úprav fotografií, které by se mohli hodit při implementaci nových funkcí fotoalba, proto padla volba na něj. Přístup k tomuto softwaru z Ruby nám poskytne gem balíček rmagick[10]. V době psaní tohoto textu je již k dispozici verze 2.0, avšak v mém projektu je použita verze 1.9. V náhledu fotoalba se objeví vždy osm malých fotografií ve dvou řadách. Je tomu tak z důvodu optimalizace pro rozlišení obrazovky 1024x768, jež ještě stále mnoho uživatelů internetu používá. Vzhledem k paměťové nenáročnosti těchto malých náhledů jsem se je rozhodl prostřednictvím modulu rmagick generovat a ukládat již při samotném nahrávání originálního snímku na server. Posílání snímků v originální velikosti by vedlo ke zbytečně velkým datovým tokům a pomalému nahrávání každé stránky náhledu fotografií. Generování osmi náhledů až při prohlížení alba by zase zbytečně zatěžovalo webový server. Naproti tomu v případě náhledu na konkrétní fotografii je tento již poněkud většího formátu. Proto je generování tohoto snímku odloženo až do okamžiku prohlížení uživatelem, kdy se nejdříve tento náhled v příslušné velikosti vytvoří do dočasného souboru, který se pošle jako výsledný obrázek prohlížeči. Pokud je součástí fotografie informace o její orientaci, album ji samo nastaví na tu správnou. Těsně před dokončením byly do tohoto projektu ještě přidány funkce na otáčení fotografie dle přání uživatele. Toto však hlavně slouží k demonstraci možností knihovny rmagick k dalšímu rozvoji fotoalba dle přání a zkušenosti uživatelů. V budoucnu by totiž mohla přibýt i možnost přímé editace.

6.11 Technologie SEO

SEO, neboli search optimalization, je v současné době velice módní záležitost. Různé komerční firmy se předhánějí v tom, která dostane webové stránky svých klientů v nejznámějších vyhledávačích výše. Při implementaci mého projektu byly mé možnosti aplikace SEO technologie velice omezené. Největší problém je absence stálého webového serveru s pevnou adresou. Tím pádem bylo nesmyslné uvažovat o tom, jak na sebe nechat odkazovat co nejvíce stránek cizích, či nechat se nějakým způsobem co nejvíce navštěvovat. Omezil jsem se tedy na základní SEO principy. Těmi jsou především hojné používání klíčových frází, a to hlavně v html značkách jakými jsou titulek stránky, nadpisy a strongy. Důležité je mít v těchto značkách nejen popis webových stránek jako takových. Především je třeba si zjistit, pod jakými dotazy by mohli potencionální zákazníci právě náš produkt hledat, a ty pak na stránce často zmiňovat. Slovem často se nerozumí pořád, ale je vhodné méně používat zájmena, ať se v našem případě slovo fotoalbum vyskytuje na úvodní stránce aspoň třikrát. O tom že by pro SEO optimalizovaná stránka měla být validní je snad i zbytečné se blíže rozepisovat. Informace k této kapitole jsem čerpal z [11].

7 Popis výsledného produktu

K spuštění této webové aplikace je nutné mít nainstalovaný jazyk ruby. Na ten postupně přidáme příkazem `gem install` následující balíčky: *gettext, mongrell, rails, rmagick, sqlite3, will_paginate*. Dále je k modulu *Rmagick* nutné doinstalovat příslušnou verzi *ImageMagicku*. Tímto by měl být server připraveen ke spuštění příkazem `ruby script/server start`.

Nyní popíši uživatelské prostředí výsledného produktu. Po zadání příslušné webové adresy se dostaneme na logovací stránku. Hned pod formulářem pro přihlášení se nachází odkaz k registraci nového uživatele. Na ten klikneme, vyplníme přihlašovací jméno, heslo, email, preferovaný jazyk a vzhled aplikace. Potvrzením formuláře byla naše registrace úspěšně provedena a je ten pravý čas se poprvé přihlásit.

Po přihlášení spatříme základní menu aplikace. To je tvořeno minimálním počtem ovládacích prvků. Kvůli zachování přehlednosti byla plocha ve webovém prohlížeči rozdělena do tří sekcí. Horní lišta poskytuje informace o aktuálně přihlášeném uživateli, prohlíženém albumu a přidělených právech k němu. Následují ještě ovládací prvky pro změnu jazyka a odhlášení.

Levá lišta obsahuje menu s základními odkazy pro snadné a rychlé ovládání aplikace. V ní se na prvním místě nachází odkaz pro náhled vašich alb. Hned pod ním pak je seznam odkazů na alba, jejichž jste vlastníkem. Pokud již máte některé album z nabídky vybráno, objeví se rychlý odkaz na přidání nové fotografie. Kliknutím níže si můžete prohlédnout alba veřejná. Následuje tlačítko pro správu práv ostatních uživatelů na vaše fotografie a jako poslední je umístěno tlačítko pro správu nastavení prostředí uživatele.

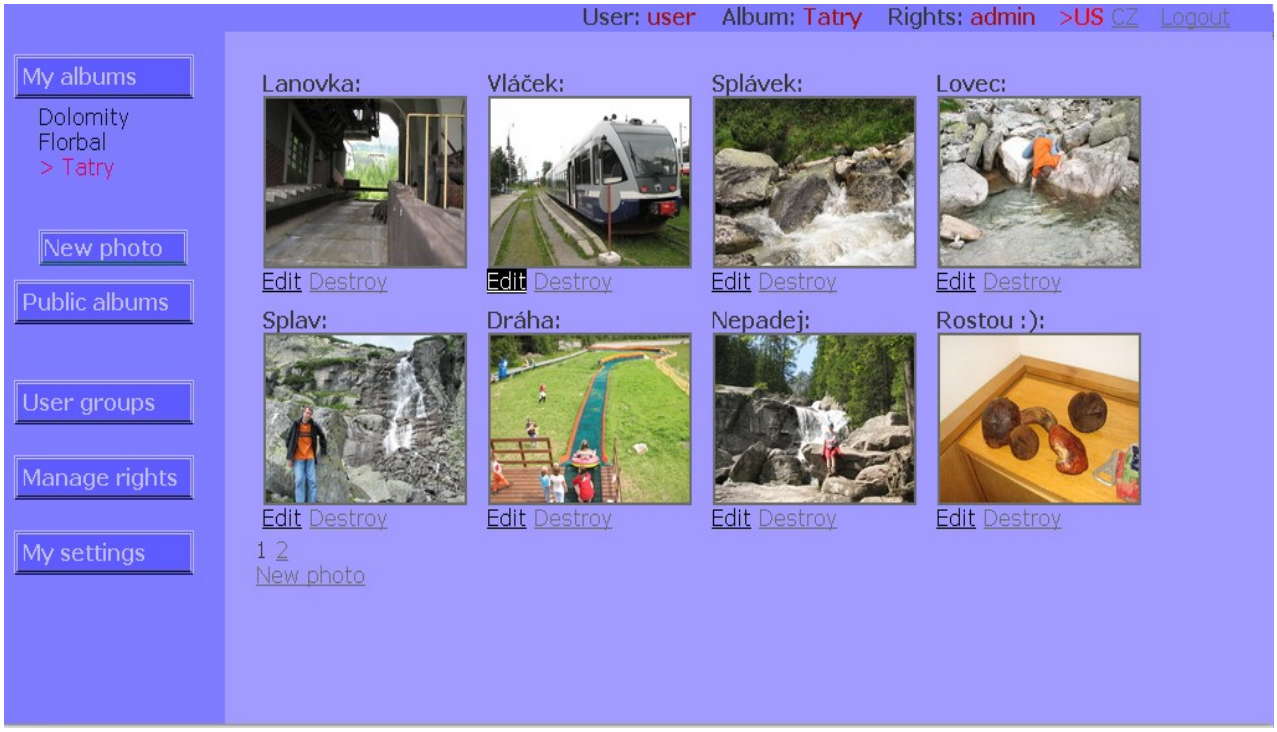
V hlavní části obrazovky se pak odehrávají veškeré ostatní činnosti. Jsou to náhledy alb vlastních i cizích, prohlížení fotografií v nich obsažených, editace práv ostatních uživatel, editace a přidávání komentářů, editace názvů alb a fotografií a mnoho dalších.

Jak je vidět z předchozího, veškeré ovládací prostředky jsou implementovány s důrazem na jdnoduchou orientaci v nich. Jedinou poněkud nepřehlednější operací, kterou bude uživatel provádět, je přidělování práv ostatním uživatelům systému. Přestože si nemyslím, že by toto ovládání někdo nezvládl, bude i ono předmětem dalších vylepšení, nejlépe na základě podnětů ze strany uživatelů.

7.1 Obrázky aplikace

Obrázek 7.1. Graf případů užití

Obrázek 7.2. Graf případů užití



Obrázek 7.3. Graf případů užití

Obrázek 7.4. Graf případů užití

8 Závěr

Výsledkem této práce je plně funkční webová aplikace fotoalba. Toto lze vzhledem k implementaci v plně přenositelném frameworku RoR nasadit prakticky na jakémkoliv operačním systému, který je v sočasnosti běžně používán. Produkt byl testován a je plně funkční v dnes nejběžněji používaných webových prohlížečích InternetExplorer7 a Mozila Firefox3.

Při tvorbě této aplikace jsem získal užitečné zkušenosti s prací ve framevorku Ruby on Rails. Kromě tohoto jsem si prohloubil znalosti v oblasti vývoje webových stránek s využitím jazyka html a kaskádových stylů. Díky provázanosti s databází jsem si rovněž zopakoval znalosti z této oblasti informačních systémů. Avšak za největší osobní přínos této práce považuji bližší seznámení s jazykem Ruby. Tento jazyk se hodně liší od jiných, které jsem dosud poznal a člověku chvíli trvá než mu přijde na chuť, ale v momentě kdy mu trochu navykne, se jen těžko vrací ke „starým“ a „zbytečně zdlouhavým“ programátorským postupům klasických programovacím jazykům.

Dalším poznatkem, který jsem si upevnil je, že pod Open Source licencí mohou vnikat stejně kvalitní produkty, jako je tomu u komerčních produktů. V případě RoR bych dokonce řekl, že stejně kvalitní prostředek pro vývoj webové aplikace budeme i na komerčním trhu hledat marně.

Při vlastní implementaci produktu vzniklo i pár problémů, na které bylo třeba pružně reagovat. Tím největším a zároveň bohužel do této chvíle přetrvávajícím je neschopnost programu msgmerge pracovat pod mým operačním systémem Windows Vista Bussines. To má za následek pád rake tasku na plnění slovníků novými slovy z aplikace. Tento problém lze obejít manuálním zkopírováním vzoru .pot do příslušné složky překladu jazyka a jeho přejmenováním na .po soubor. Poté již stačí přeložit, spustit rake task makemo a aplikace by měla být přeložena. Dalším problémem bylo špatné pozicování objektů v prostředí Internet Exploreru. To k mému údivu spravilo použití stylu „zoom: 1“. Další problémy pak již patřily do kategorie běžných a k jejich vyřešení zpravidla stačila chvíle pátrání na googlu, či jiném vyhledávači.

Výsledný produkt hodlám nadále zdokonalovat a používat pro sdílení fotografií mezi přáteli a rodinou. Možností k rozšíření funkcionality je celá řada. Vzhledem k použitému modulu rmagick není žádný problém u snímků zobrazovat EXIF informace v nich obsažené, což se mi momentálně nezdá být příliš užitečné. Mnohem zajímavější mi připadá myšlenka možnosti přímé editace nahraných fotografií na serveru. Funkce na otáčení fotografie, zvětšení a zmenšení kontrastu či světlosti a mnohé další jsou prostřednictvím modulu rmagick jednoduše dostupné, stačí tedy k nim doimplementovat příslušné rozhraní do aplikace.

Mám-li shrnout hlavní přednosti mého webového fotoalba proti ostatním, musím začít unikátností. Opravdu je pro mě důležité, aby byly mé fotografie na mých vlastních webových stránkách, což podtrhne i jejich jedinečnost. Fotoalbum chci používat také jako jeden z prostředků archivace svých nejlepších snímků, proto je pro mě důležité, že nijak neomezuje velikost fotografie

pro její nahrání na web. Další výhodou například proti serveru rajce.net je možnost přídělovat práva na své fotografie známým, kolegům, rodinným příslušníkům a podobně. Těchto je navíc několik typů, takže každému přidělíte dle vlastní libosti určitý stupeň práv pro konkrétní fotoalbum. Další výhodou je možnost přizpůsobení vzhledu aplikace požadavkům uživatele prostřednictvím předem nadefinovaných skinů.

Vzhledem k mým malým zkušenostem s tvorbou webu a slabému grafickému cítění je pak pro někoho možná nevýhodou příliš strohý a jednoduchý vzhled. Na druhou stranu však prostředí neruší přílišnými kontrasty v prohlížení toho hlavního – fotografií.

Literatura

[5] doc. Ing. Jaroslav Zendulka, Ing. Vladimír Bartík, Ph.D., Ing. Šárka Květoňová . Anaĺýza a návrh počítačových systémů – studijní opora. 2006 [cit. 2008-07-27].

[6] Steven Holzner. Začínáme programovat v Ruby on Rails. Computer press 2007.

[7] Kacper Cieśla, Comboy . User authentication in Ruby on Rails [online]. 2007 [cit. 2008-07-27]. Dostupný z WWW: <http://codingbitch.com/p/comboy/User+authentication+in+Ruby+on+Rails>.

[8] Václav Slavík, About Poedit [online]. 2008 [cit. 2008-07-27]. Dostupný z WWW: <http://www.poedit.net/>.

[9] Fuji Hawk Eggplant . Ruby-GetText-Package HOWTO for Ruby on Rails [online]. 2008 [cit. 2008-07-27]. Dostupný z WWW: <http://www.yotabanana.com/hiki/ruby-gettext-howto-rails.html>.

[10] Timothy P. Hunter . About RMagick [online]. 2008 [cit. 2008-07-27]. Dostupný z WWW: <http://rmagick.rubyforge.org/>.

[1] Timothy P. Hunter . Image:Model view controller.png [online]. 2008 [cit. 2008-07-31]. Dostupný z WWW: <http://www.wikimedia.org/>.

[2] Rajče.net – místo pro vaše fotografie <http://www.rajce.net/>.

[3] Fkickr – Photo Sharing [online]. [cit. 2008-07-31]. <http://www.wikimedia.org/>.

[4] ImageShack – online media hosting [online]. [cit. 2008-07-31]. <http://www.ImageShack .us/>

[5] doc. Ing. Jaroslav Zendulka, Ing. Vladimír Bartík, Ph.D., Ing. Šárka Květoňová . Anaĺýza a návrh počítačových systémů – studijní opora. 2006 [cit. 2008-07-27].

[6] Steven Holzner. Začínáme programovat v Ruby on Rails. Computer press 2007.

[7] Kacper Cieśla, Comboy . User authentication in Ruby on Rails [online]. 2007 [cit. 2008-07-27]. Dostupný z WWW: <http://codingbitch.com/p/comboy/User+authentication+in+Ruby+on+Rails>.

[8] Václav Slavík, About Poedit [online]. 2008 [cit. 2008-07-27]. Dostupný z WWW: <http://www.poedit.net/>.

[9] Fuji Hawk Eggplant . Ruby-GetText-Package HOWTO for Ruby on Rails [online]. 2008 [cit. 2008-07-27]. Dostupný z WWW: <http://www.yotabanana.com/hiki/ruby-gettext-howto-rails.html>.

[10] Timothy P. Hunter . About RMagick [online]. 2008 [cit. 2008-07-27]. Dostupný z WWW: <http://rmagick.rubyforge.org/>.

[11] Jaimie Sirovich and Cristian Darie . Professional Search Engine Optimization with PHP: A Developer's Guide to SEO . Wrox . 2007 .

Příloha – Obsah přiloženého CD

Na přiloženém CD najdete kompletní zdrojové kódy vícejazyčného fotoserveru v Ruby on Rails ve dvou provedeních. Ve složce „Prazdna“ je připravena čerstvě zmigrovaná, zatím aplikace k instalaci na server. V adresáři „Naplnena“ je pak projekt se vzorkem testovacích alb, fotografií a uživateli „*admin*“, „*user*“ a „*cesky*“. Všichni mají heslo stejné, jako svoje přihlašovací jméno. Součástí jsou i metadata pro vývojove prostředí Aptana Studio. Pro spuštění serveru je nutné mít nainstalován jazyk Ruby se všemi potřebnými gem balíčky. Tyto jsou vyjmenovány níže. Dále je třeba mít nainstalovánu aplikaci ImageMagick pro správnou funkci gem modulu rmagick. Součástí CD je i kompletní technická dokumentace včetně zadání a licenční smlouva.

Potřebné gem balíčky: gettext, mongrell, rails, rmagick, sqlite3, will_paginate